



# Calorimetry Software Meeting 14/06/02

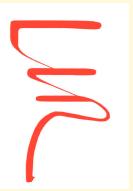


## Toward CaloDataFrame Persistence

David Chamont & Pascal Paganini



# Interest



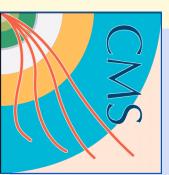
- ◆ Better separation between simulation and reconstruction
- ◆ Allow to read real data from testbeam
- ◆ Closer to the real scheme we will have with raw data

# Software consequences (1)



- ◆ New package in Calorimetry:  
`Calorimetry/CaloDataFrame`
- ◆ Containing:
  - `CaloDataFrame(.dll)`
  - `CaloDataFrameFormatter`
  - `CaloDataFramePERSISTANTSetup`
  - `CaloDataFrameROUFactory`
- ◆ Old `CaloReadout/ ... /CaloDataFrame` removed

## Software consequences (2)



- ◆ Usage of calo iterators Instead of CaloFrontEndResponse::GetCache() :

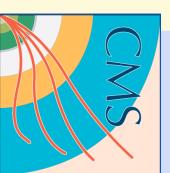
```
Caloltr<CaloDataFrame> frames("CDEBRY01");
Caloltr<CaloDataFrame> frames("CDFFRY01");
Caloltr<CaloDataFrame> frames("CDESFX01");
Caloltr<CaloDataFrame> frames("HDHCAL01");
```

- ◆ Need changes in:

- CaloReadout: CaloFrontEndResponse, CaloVCoder
- EcalRealistic: EcalRealisticCoder, EcalRealisticFrontEnd
- HcalRealistic: HcalRealisticCoder, HcalRealisticFrontEnd
- PreshRealistic: PreshRealisticCoder, PreshRealisticFrontEnd
- EcalDetailedSim: EdsFrontEndResponse, EdsFrontEnd
- EcalDetailedRec: EdrReconstructor

- ◆ CaloDataFrame changes:

- CaloDataFrame contains:
  - ↳ Nmax samples (integers)
  - ↳ CellID
- Nmax must be fixed (not dynamic) for persistency in objectivity
  - ↳ vector<int> □ int[Nmax]
  - ↳ Nmax = 10

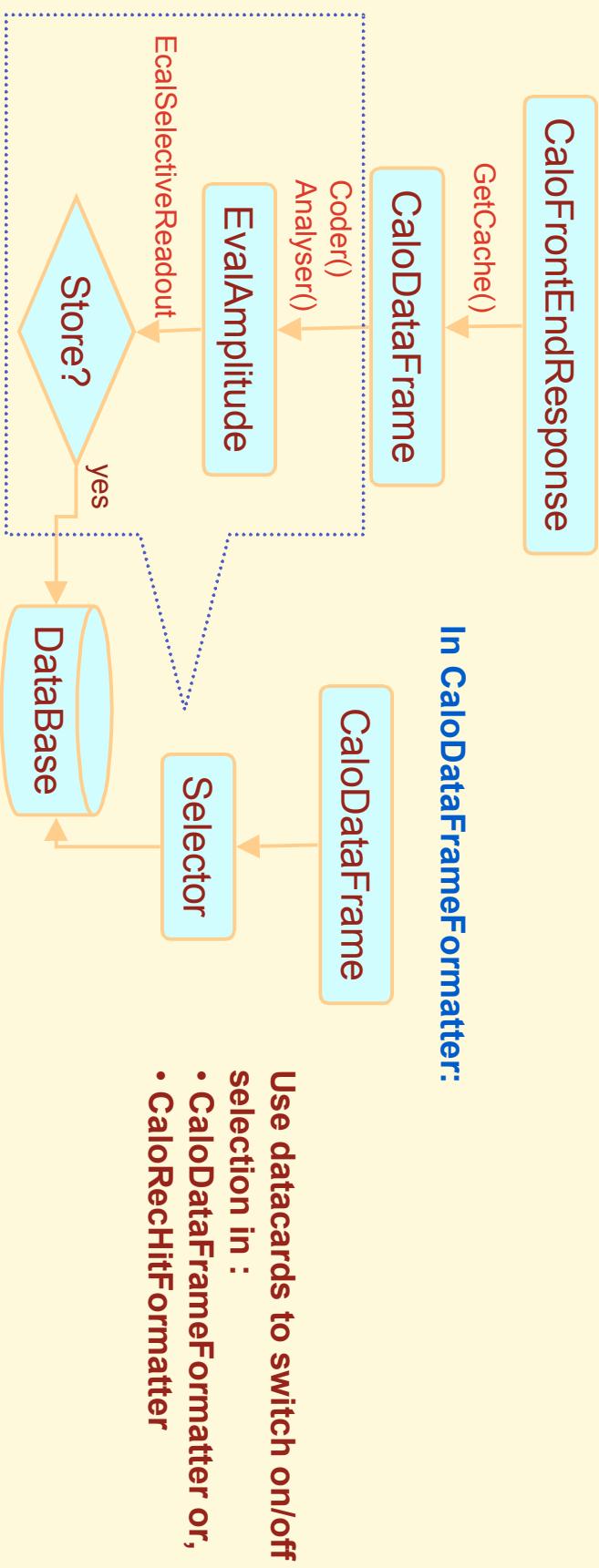


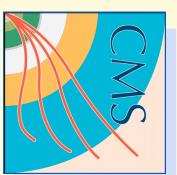
## Data Volume



- ◆ Data volume too large □ need selection
- ◆ keep only CaloDataFrame coming from real hits?
  - No: At high lumi, ~50% of the Ecal is occupied
- ◆ Apply a selective readout at the CaloDataFrame level?

In CaloRecHitFormatter:





# How to tell CaloDataFrame the way it has been coded?



CaloDataFrameWriter

CaloDataFrameReader

simHits

rechits

CaloTimeSamples

CaloDataFrames

Coder

Coder

Depends on the  
digitization lib

Same lib?

## ♦ Coder new attribute of CaloDataFrame?

- no, too heavy in memory ...

## ♦ Trust in the user but ... check with a "magic DataFrame" ?

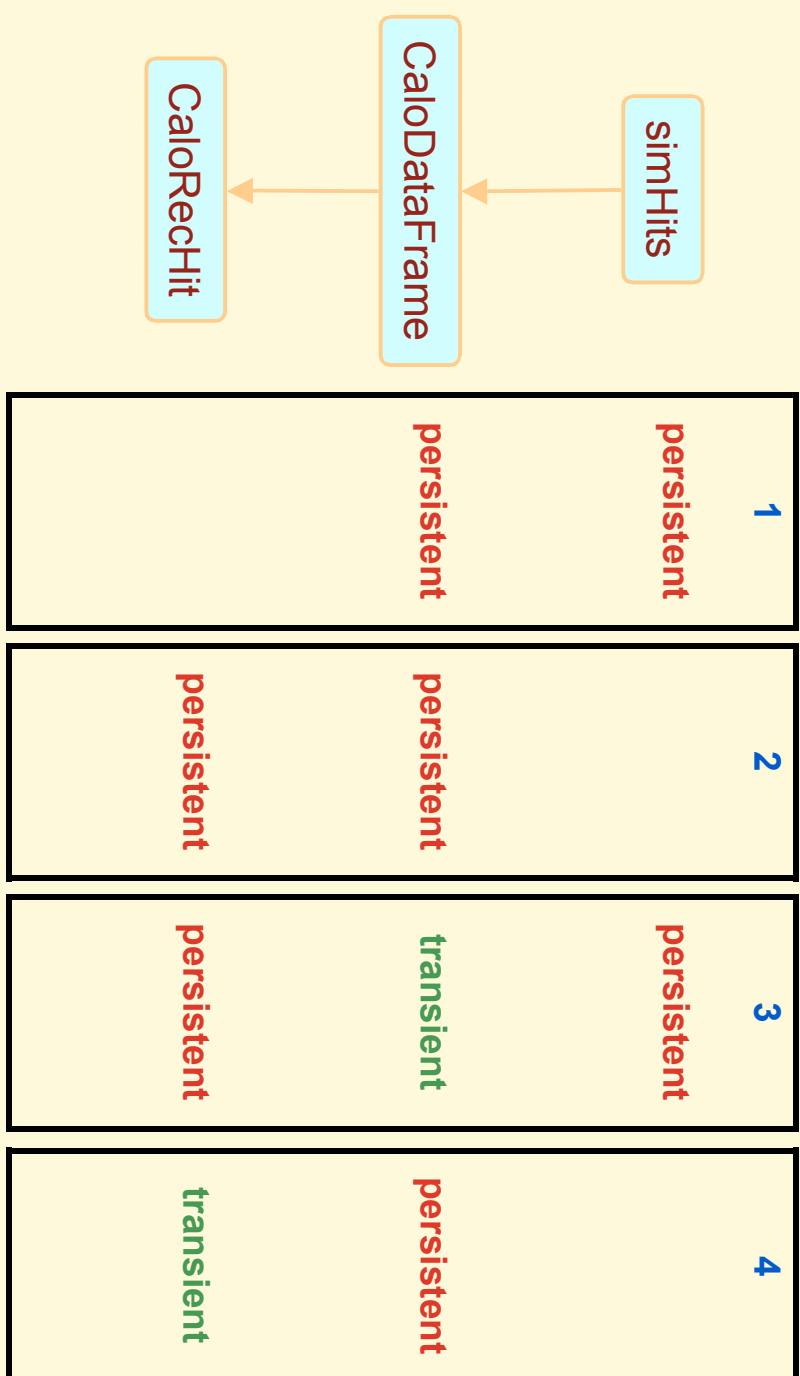
- the first CaloDataFrame of a container has a special coding depending on the coder used

```
In CaloDataFrameWriter:  
CaloDataFrame magic = coder.Magic();  
if (first) _cache.push_back(magic);  
If (first) {  
    if (magic==(*frames)) // ok ;
```

```
In CaloDataFrameReader:  
CaloItr<CaloDataFrame> frames;  
CaloDataFrame magic = coder.Magic();  
If (first) {
```



## Use cases



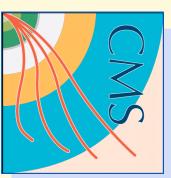
**1 = CaloDataFrameWriter**

**2 = CaloDataFrameReader + RecHitsWriter**

**3 = current production behaviour (RecHitsWriter through transient CaloDataFrames)**

**4 = CaloDataFrameReader**

# Release plan



- ◆ **Step 1:**

- First commit when the new code will produce EXACTLY the same behaviour as in the current release: use case 3
- Need to be able to make CaloRecHit persistent and CaloDataFrame transient
- Problem: in COBRA, GoPersistent =1 stores in the database all persistent data (CaloRecHit and CaloDataFrame are both stored)
- Wait for COBRA\_6\_2\_0: new possibility: XX:Request = Transient

- ◆ **Step 2:**

- Re-implemented the selective readout at the level of CaloDataFrame
- Work in coordination with Scott

- ◆ **Step 3:**

- Coder....